Library Extension Assignment

DT228/DT282
Software Engineering 2

Shane Buckley
C20703429

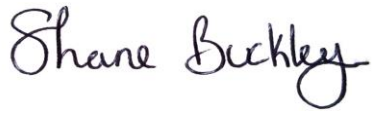
School of Computer Science
Technological University Dublin

04/05/2022

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:



_____

Shane Buckley

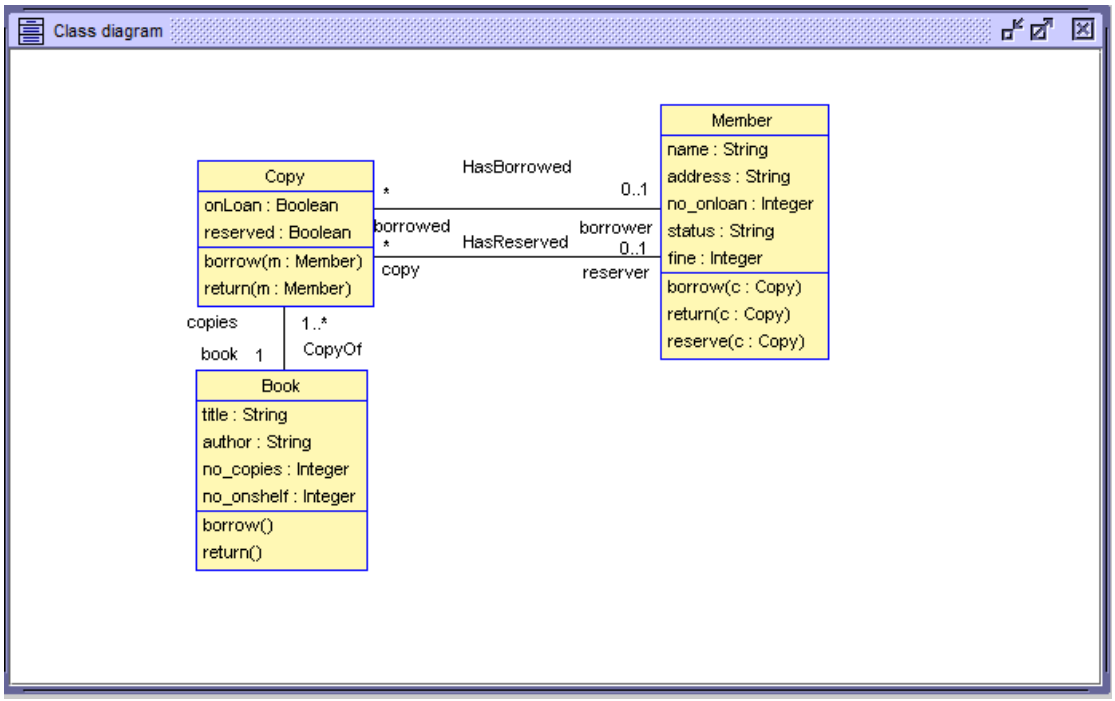
04/05/2022

Table of Contents

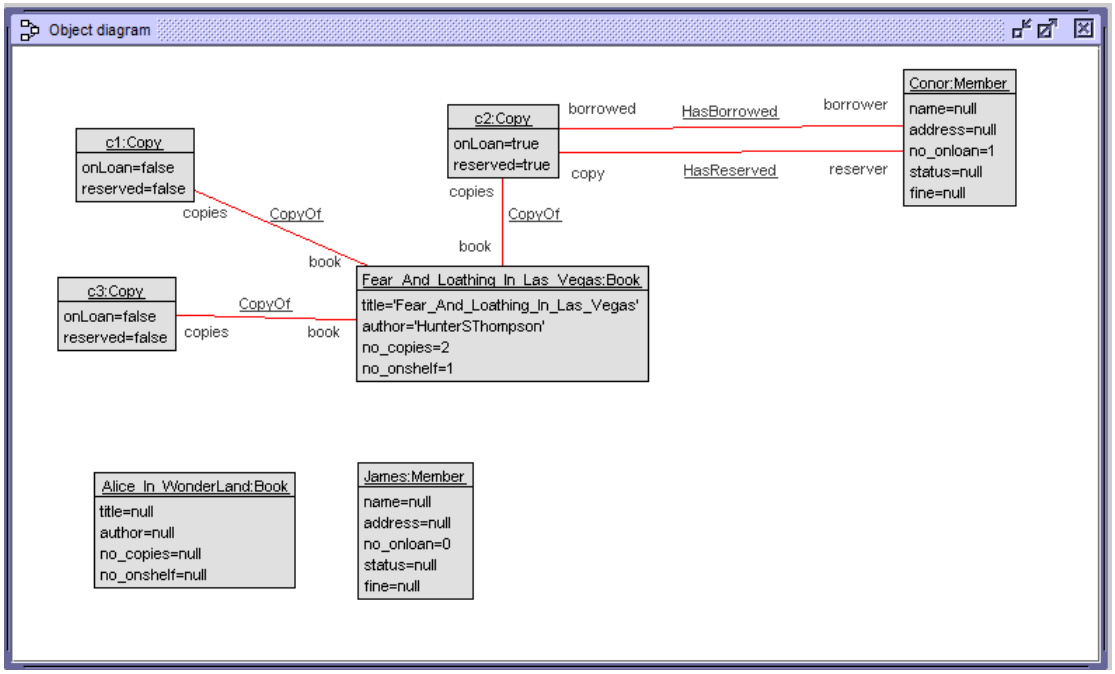
1. Overview

For this assignment I chose the “Extend and test a more comprehensive USE model for the Library system in USE” option. The use case I have developed for the Library system is the reserve book use case and includes constraints to ensure that when a book is being reserved, that it hasn’t been reserved or borrowed already, and also if a member has reserved a book that they are trying to borrow. I have implemented statemachines to handle the status of a copy. Included with my code is a SOIL file containing some sample SOIL commands to create member, book and copy objects for testing and demonstration purposes.

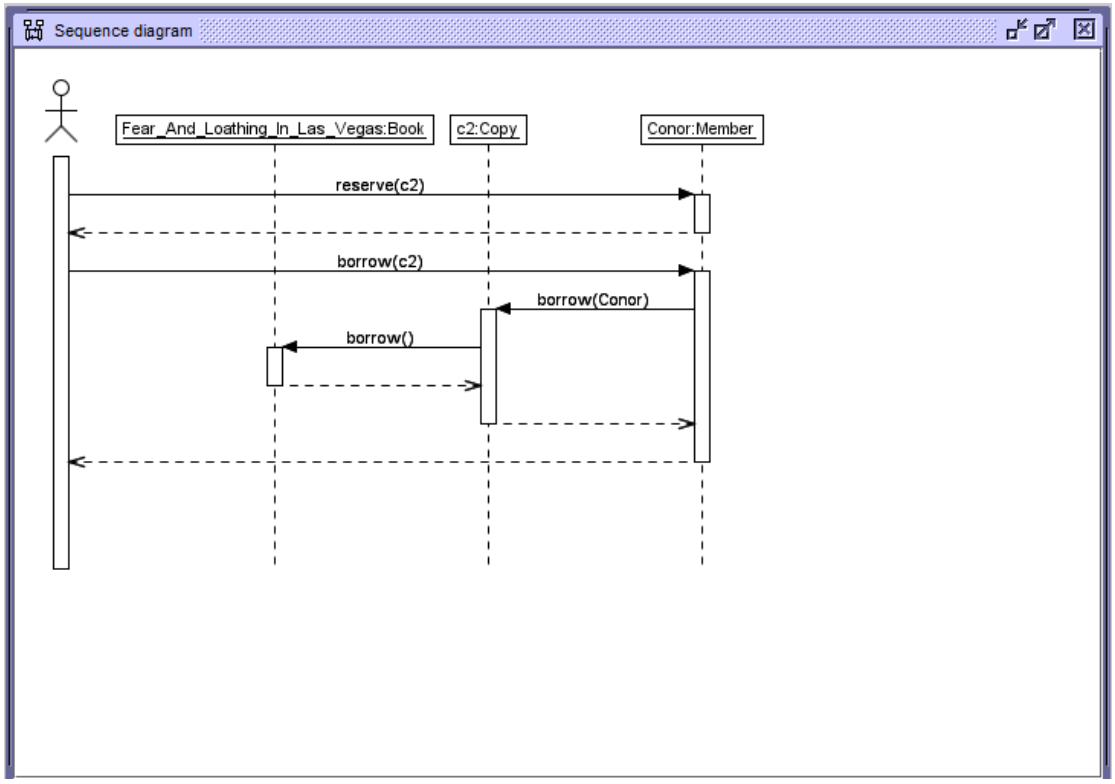
2. Class Diagram

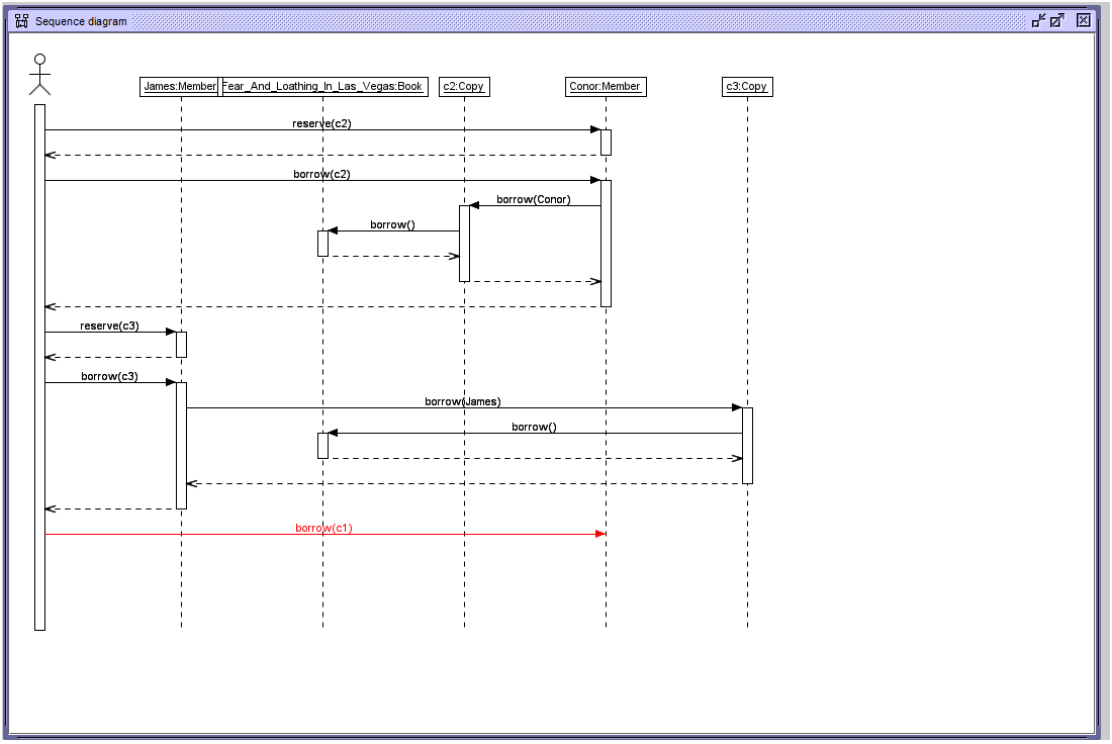


3. Object Diagram



4. Sequence Diagrams





5. Constraints

```
constraints


context Member::return(c:Copy)
    pre: c.onLoan = true
    pre: self.borrowed->includes(c)

context Member::reserve(c:Copy)
    pre: c.reserved = false
    pre: c.onLoan = false
    post: self.copy->includes(c)
    post: c.reserved = true

context Member::borrow(c:Copy)
    pre limit: self.no_onloan < 1
    pre: self.borrowed->excludes(c)
    pre: c.reserved = false or self.copy->includes(c)
    pre: c.onLoan = false
    post: c.onLoan = true
    post: self.borrowed->includes(c)
```

6. Testing

When a member attempts to reserve a book that has already been reserved by another member:

```
Lib.soil>
use> !James.reserve(c3)
use> !Conor.reserve(c3)
[Error] 1 precondition in operation call `Member::reserve(self:Conor, c:c3)' does not hold:
  pre3: (c.reserved = false)
    c : Copy = c3
    c.reserved : Boolean = true
    false : Boolean = false
    (c.reserved = false) : Boolean = false

  call stack at the time of evaluation:
    1. Member::reserve(self:Conor, c:c3) [caller: Conor.reserve(c3)@<input>:1:0]

+-----------------------------------------------------------------+
| Evaluation is paused. You may inspect, but not modify the state. |
+-----------------------------------------------------------------+

Currently only commands starting with `?', `:', `help' or `info' are allowed.
`c' continues the evaluation (i.e. unwinds the stack).
```

A member successfully borrowing a book they have reserved:

```
Lib.soil>
use> !James.reserve(c3)
use> !James.borrow(c3)
use>
```

A member trying to borrow a book they have not reserved:

```
use> !Conor.borrow(c1)
[Error] 1 precondition in operation call `Member::borrow(self:Conor, c:c1)' does not hold:
  limit: (self.no_onloan < 1)
    self : Member = Conor
    self.no_onloan : Integer = 1
    1 : Integer = 1
    (self.no_onloan < 1) : Boolean = false

  call stack at the time of evaluation:
    1. Member::borrow(self:Conor, c:c1) [caller: Conor.borrow(c1)@<input>:1:0]

+-----------------------------------------------------------------+
| Evaluation is paused. You may inspect, but not modify the state. |
+-----------------------------------------------------------------+

Currently only commands starting with `?', `:', `help' or `info' are allowed.
`c' continues the evaluation (i.e. unwinds the stack).

>
```

7. Source Code

StateMachines.use

```
model Library

class Book
  attributes
    title : String
    author : String
    no_copies : Integer
    no_onshelf : Integer
  operations
    borrow()
    begin
        self.no_onshelf := self.no_onshelf - 1
    end
    pre copiesOnShelf: no_copies >0
    post: no_onshelf = no_onshelf@pre - 1

    return()
    begin
        self.no_onshelf := self.no_onshelf + 1
    end

    reserve()
    begin
        self.no_onshelf := self.no_onshelf - 1
    end

end

class Copy
  attributes
    status : String
  operations
    borrow( m : Member)
    begin
        self.status := 'onLoan';
        self.book.borrow()
    end
    return( m : Member)
    begin
        self.status := 'onShelf';
        self.book.return()
    end
    reserve( m : Member)
    begin
```

```
        self.status:= 'isReserved';
        self.book.reserve();
    end


  statemachines
    psm States
    states
        newCopy : initial
        available   [status = 'onShelf']
        taken       [status = 'onLoan']
        reserved    [status = 'isReserved']
    transitions
        newCopy -> available { create }
        available -> taken  { borrow() }
        available -> reserved { reserve() }
        reserved -> taken { borrow() }
        taken -> available { return() }
    end
end

class Member
  attributes
    name : String
    address : String
    no_onloan : Integer
    status : String
    fine : Integer
  operations
    borrow(c : Copy)
    begin
        insert (self, c) into HasBorrowed;
        self.no_onloan := self.no_onloan + 1;
        c.borrow(self);
    end


    return( c: Copy)
    begin
        self.no_onloan := self.no_onloan - 1;
        c.return(self);
        delete (self, c) from HasBorrowed;
    end

    reserve( c: Copy)
    begin
        insert (self, c) into HasReserved;
        c.reserve(self);
    end
```

```
end


association HasBorrowed between
    Member[0..1] role borrower
    Copy[*] role borrowed
end

association CopyOf between
    Copy[1..*] role copies
    Book[1] role book
end

association HasReserved between
    Member[0..1] role reserver
    Copy[*] role copy
end

constraints

context Member::borrow(c:Copy)
    pre limit: self.no_onloan < 1
    pre: self.borrowed->excludes(c)
    pre: c.status = 'onShelf' or self.copy->includes(c)
    post: c.status = 'onLoan'
    post: self.borrowed->includes(c)


context Member::reserve(c:Copy)
    pre: c.status = 'onShelf'
    post: self.copy->includes(c)
    post: c.status = 'isReserved'

context Member::return(c:Copy)
    pre: c.status = 'onLoan'
    pre: self.borrowed->includes(c)
    post: c.status = 'onShelf'
```

Lib.use

```
model Library

class Book
  attributes
    title : String
    author : String
```

```
    no_copies : Integer
    no_onshelf : Integer
  operations
    borrow()
    begin
        self.no_onshelf := self.no_onshelf - 1
    end
    pre copiesOnShelf: no_copies >0
    post: no_onshelf = no_onshelf@pre - 1

    return()
    begin
        self.no_onshelf := self.no_onshelf + 1
    end

end

class Copy
  attributes
    onLoan : Boolean
    reserved : Boolean
  operations
    borrow( m : Member)
    begin
        self.onLoan := true;
        self.book.borrow()
    end
    return( m : Member)
    begin
        self.onLoan := true;
        self.book.return()
    end
end

class Member
  attributes
    name : String
    address : String
    no_onloan : Integer
    status : String
    fine : Integer
  operations
    borrow(c : Copy)
    begin
        insert (self, c) into HasBorrowed;
        self.no_onloan := self.no_onloan + 1;
        c.borrow(self);
    end
```

```
    return( c: Copy)
    begin
        self.no_onloan := self.no_onloan - 1;
        c.return(self);
        delete (self, c) from HasBorrowed;
    end

    reserve( c: Copy)
    begin
        insert (self, c) into HasReserved;
        c.reserved:= true;
    end

end

association HasReserved between
    Member[0..1] role reserver
    Copy[*] role copy
end

association HasBorrowed between
    Member[0..1] role borrower
    Copy[*] role borrowed
end

association CopyOf between
    Copy[1..*] role copies
    Book[1] role book
end



constraints


context Member::return(c:Copy)
    pre: c.onLoan = true
    pre: self.borrowed->includes(c)

context Member::reserve(c:Copy)
    pre: c.reserved = false
    pre: c.onLoan = false
    post: self.copy->includes(c)
    post: c.reserved = true

context Member::borrow(c:Copy)
    pre limit: self.no_onloan < 1
```

```
    pre: self.borrowed->excludes(c)
    pre: c.reserved = false or self.copy->includes(c)
    pre: c.onLoan = false
    post: c.onLoan = true
    post: self.borrowed->includes(c)
```

Lib.soil

```
-- Script generated by USE 4.1.1

!new Member('James')
!James.no_onloan := 0

!new Book('Fear_And_Loathing_In_Las_Vegas')
!Fear_And_Loathing_In_Las_Vegas.title :=
'Fear_And_Loathing_In_Las_Vegas'
!Fear_And_Loathing_In_Las_Vegas.author := 'HunterSThompson'

!new Copy('c1')
!c1.reserved := false
!c1.onLoan := false
!insert(c1,Fear_And_Loathing_In_Las_Vegas) into CopyOf

!new Copy('c2')
!c2.onLoan := false
!c2.reserved := false
!insert(c2,Fear_And_Loathing_In_Las_Vegas) into CopyOf
!insert (c1,Fear_And_Loathing_In_Las_Vegas) into CopyOf
!insert (c2,Fear_And_Loathing_In_Las_Vegas) into CopyOf
!Fear_And_Loathing_In_Las_Vegas.no_copies := 2
!Fear_And_Loathing_In_Las_Vegas.no_onshelf := 2

!new Member('Conor')
!Conor.no_onloan := 0

!new Copy('c3')
!c3.reserved := false
!c3.onLoan := false
!insert(c3,Fear_And_Loathing_In_Las_Vegas) into CopyOf

!new Book('Alice_In_WonderLand')
!WandP.author := 'LewisCarroll'
!WandP.title := 'Alice_In_WonderLand'

!Conor.reserve(c2)
--!James.reserve(c2)
--!James.borrow(c2)
!Conor.borrow(c2)
```